Uncertainty Quantification for Robust Design

Régis Duvigneau, Massimiliano Martinelli, Praveen Chandrashekarappa

Uncertainty Quantification for Robust Design

The objective of this chapter is to present, analyze and compare some practical methods that could be used in engineering to quantify uncertainty, for mechanical systems governed by partial differential equations. Most applications refer to aerodynamics, but the methods described in this chapter can be applied easily to other disciplines, such as structural mechanics.

0.1. Introduction

Simulation-based performance prediction has been an active research topic for many years and is now applied for industrial test-cases in all fields of computational mechanics. However, classical methods developed in Computational Fluid Dynamics (CFD) or Computational Structural Mechanics (CSM) usually assume a perfect knowledge of the parameters of the system studied, such as geometry, operational conditions, etc. Under this asumption, efficient numerical methods have been developed, yielding an accurate performance prediction for optimization purpose. However, everyday life is subject to uncertainty and the parameters of every systems are subject to random fluctuations. For instance, the flight conditions of an aircraft can fluctuate according to the weather, the wing design can vary because of manufacturing tolerances, etc. These uncertainties modify the response of the system and in some cases significantly degrade its performance. This is particularly true if the system has been optimized for some precise operational conditions.

Therefore, there is a growing interest for *robust design* methods, that takes into account uncertainty during the design phase. The objective of such approaches is to determine a design which has a satisfactory performance at nominal operational

---

Chapter written by Régis DUVIGNEAU, Massimiliano MARTINELLI and Praveen CHANDRASHEKARAPPA.

conditions and a reasonable performance degradation when these conditions fluctuate. Most of these methods, described in a previous chapter, rely on the estimation of statistics of the performance, such as expectation and variance. However, these statistical quantities are not straightforward to compute for systems governed by partial differential equations. Indeed, uncertain parameters are input variables for the simulation code, whereas the performance is an output variable. Therefore, the estimation of performance statistics rely on *the propagation of uncertainty* through the simulation code.

The methods for uncertainty quantification require a knowledge of the performance behavior in the vicinity of the nominal conditions. In the framework of systems governed by partial differential equations, they are based on the construction of a simplified model. In this chapter, we present and compare two practical methods that rely on two different points of view:

- The *method of moments* considers as simplified model the Taylor series expansion of the performance with respect to the uncertain parameters, around the nominal conditions. This linear or quadratic model, constructed from one point only, is integrated analytically to provide statistics of the performance ;

- The *metamodel-based Monte-Carlo methods* consider as simplified model a metamodel that interpolate several points stored in a database. The statistics of the performance are estimated thanks to a Monte-Carlo integration on the basis of the metamodel.

These two approaches are significantly different, since the former computes the system performance at the nominal conditions only but requires also derivatives, whereas the latter computes several performance values at several conditions.

In this chapter, we first describe the two approaches, in a general framework, and then compare them in the context of aerodynamics.

0.2. Problem statement

We consider that the performance of the system can be written as a functional $j(\gamma) = J(\gamma, W) \in \mathrm{R}$, where $\gamma \in \mathrm{R}^n$ are parameters that define the operational conditions and the state variables $W = W(\gamma) \in \mathrm{R}^{N_S}$ satisfy a (nonlinear) state equation:

$$\Psi(\gamma, W) = 0. \tag{1}$$

This formulation can be applied to several engineering problems governed by partial differential equations, such as fluid dynamics, structural mechanics or electromagnetics. For instance, one would like to minimize the aerodynamic resistance (drag coefficient) of an aircraft, minimize the weight or the compliance of a structure, maximize

the energy transmitted by an electromagnetic wave in a given direction. The operational conditions represent inflow conditions, the geometry of the problem, materials, etc.

Suppose now that the operational conditions γ are subject to uncertainty. Then, they have to be considered as random variables characterized by a Probability Density Function (PDF) denoted f. As consequence, the performance j of the system should also be considered as a random variable, for which we would like to estimate some statistical quantities, such as expectation μ_j and variance σ_j^2:

$$\begin{aligned} \mu_j &= E\big[j(\gamma)\big] = \int j(\gamma) f(\gamma)\, d\gamma \\ \sigma_j^2 &= E\big[(j(\gamma) - \mu_j)^2\big] = \int j(\gamma)^2 f(\gamma)\, d\gamma - \mu_j^2 \end{aligned} \tag{2}$$

However, this estimation is not straightforward, since j is a *functional* that depends on states variables W.

0.3. Estimation using the method of moments

0.3.1. *Presentation of the method*

The idea behind the Method of Moments [PUT 01, BEY 07] (MoM) is based on the Taylor series expansion of the original nonlinear functional $j(\gamma)$ around the *mean value* of the uncertain variables, and then computing mean and variance of the output by using the *moments* of the distribution for the input variables.

Let us consider that the uncertain variables $\gamma = (\gamma_1, \ldots, \gamma_n)$ can be decomposed as: $\gamma = \mu_\gamma + \delta\gamma$ of a fully deterministic quantity $\mu_\gamma = E\big[\gamma\big]$ (the mean value of γ) with a stochastic perturbation $\delta\gamma$. Then, the Taylor expansion of the functional $j(\gamma)$ around μ_γ reads:

$$j(\gamma) = j(\mu_\gamma) + \sum_i G_i \delta\gamma_i + \frac{1}{2} \sum_{i,k} H_{i,k} \delta\gamma_i \delta\gamma_k + O(||\delta\gamma||^3), \tag{3}$$

where $\delta\gamma_i = \gamma_i - \mu_{\gamma_i}$, $G_i = \frac{dj}{d\gamma_i}\big|_{\mu_\gamma}$ and $H_{i,k} = \frac{d^2 j}{d\gamma_i d\gamma_k}\big|_{\mu_\gamma}$.

When we compute the expectation value of (3), the term containinig the first order derivative disappears (due to the definition of mean value, the property $\int \delta\gamma f(\gamma)\, d\gamma = 0$ holds) and the mean value of the functional $j(\gamma)$ is approximated by:

$$\mu_j \simeq j(\mu_\gamma) + \frac{1}{2} \sum_{i,k} C_{i,k} H_{i,k} \tag{4}$$

where $C_{i,k}$ is the (i,k)-element of the covariance matrix ($C_{i,k} = \int(\gamma_i - \mu_{\gamma_i})(\gamma_k - \mu_{\gamma_k}) f(\gamma)\, d\gamma$) and depends only on the statistical model (given by the PDF function $f(\gamma)$) for the uncertain variables γ. Therefore, the cost to evaluate the integral is given by the cost for computing the functional j and Hessian matrix $\frac{d^2 j}{d\gamma^2}$ both evaluated at μ_γ, the mean value of the input variables.

In the same way we can write a second-order approximation for the variance defined in (2):

$$\sigma_j^2 \simeq \sum_{i,k} C_{i,k} G_i G_k + \frac{1}{4} \sum_{i,k,l,m} \big(C_{i,l} C_{k,m} + C_{i,m} C_{k,l}\big) H_{i,k} H_{l,m}. \tag{5}$$

A more general formulation can be found in [BEY 07] or [MAR 07].

It is important to note that the equations for the mean (4) and the variance (5) require the gradient and the Hessian of the functional $j(\gamma)$, both evaluated at μ_γ: for this reason the method above is commonly known as *second-order Method of Moments*. If we get rid of the second derivatives in (4)-(5) we obtain the *first-order Method of Moments*, in which the mean value of the functional is approximated with the functional evaluated at the mean value of the uncertain variables, i.e. $\mu_j = j(\mu_\gamma)$.

Another important point is that we are using a local approximation of the functional for the estimation of a quantity that is inherently global: the integral of the functional weighted by a PDF. Therefore the accuracy of the estimation will depend on the fact that the local approximation is appropriate or not to take into account the variability of the uncertain variables. In other words, this kind of approach will be appropriate if the variability of the uncertain variables is less than the interval in which the functional can be approximated by its Taylor expansion.

Besides the difficulties pointed out in the previous comments, the most challenging task to apply the Method of Moments is the evaluation of the gradient and the Hessian of a functional constrained by a nonlinear equation (typically a set of PDEs).

0.3.2. *Computation of the derivatives*

We aim at computing the derivatives of the performance with respect to uncertain variables [GHA 06, Tay 01]. Using the chain rule, the gradient of the functional with respect to each component of γ is given by:

$$\frac{dj}{d\gamma_i} = \frac{dj}{d\gamma} e_i = \frac{\partial J}{\partial \gamma_i} + \frac{\partial J}{\partial W} \frac{dW}{d\gamma_i}. \tag{6}$$

The differentiation of the state equation reads:

$$\frac{\partial \Psi}{\partial \gamma_i} + \frac{\partial \Psi}{\partial W} \frac{dW}{d\gamma_i} = 0. \tag{7}$$

This equation yields the computation of the state sensitivities $\theta_i = \frac{dW}{d\gamma_i}$ by solving the following linear system:

$$\frac{\partial \Psi}{\partial W}\theta_i = -\frac{\partial \Psi}{\partial \gamma_i}. \tag{8}$$

The first-order derivatives of j with respect to uncertain parameters γ can be obtained by solving equation (8) to obtain the state sensitivities first, and then by using (6). However, using such a method, we should solve one linear system for each uncertain parameter γ_i. It is more efficient to adopt a so-called adjoint approach. Combining equations (6) and (7) in transposed form, we get:

$$\left(\frac{dj}{d\gamma}\right)^\top = \left(\frac{\partial J}{\partial \gamma}\right)^\top - \left(\frac{\partial \Psi}{\partial \gamma}\right)^\top \left(\frac{\partial \Psi}{\partial W}\right)^{-\top} \left(\frac{\partial J}{\partial W}\right)^\top.$$

Then, we can easily obtain the gradient of j by solving the adjoint system first:

$$\left(\frac{\partial \Psi}{\partial W}\right)^\top \Pi = \left(\frac{\partial J}{\partial W}\right)^\top, \tag{9}$$

where Π are the adjoint variables, and then by computing:

$$\left(\frac{dj}{d\gamma}\right)^\top = \left(\frac{\partial J}{\partial \gamma}\right)^\top - \left(\frac{\partial \Psi}{\partial \gamma}\right)^\top \Pi. \tag{10}$$

Using the adjoint approach, only one linear system solving is required, whatever the number of uncertain variables.

Starting from the first-order derivative (6), we perform another differentiation with respect to the k-th component of γ, which reads:

$$\frac{d^2 j}{d\gamma_i d\gamma_k} = D^2_{i,k} J + \frac{\partial J}{\partial W}\frac{d^2 W}{d\gamma_i d\gamma_k}, \tag{11}$$

with:

$$\begin{aligned} D^2_{i,k} J = &\frac{\partial}{\partial \gamma}\left(\frac{\partial J}{\partial \gamma} e_i\right) e_k + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial \gamma} e_i\right)\frac{dW}{d\gamma_k} \\ &+ \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial \gamma} e_k\right)\frac{dW}{d\gamma_i} + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial W}\frac{dW}{d\gamma_i}\right)\frac{dW}{d\gamma_k}. \end{aligned}$$

Then, we differentiate equation (7) to obtain:

$$D^2_{i,k}\Psi + \frac{\partial \Psi}{\partial W}\frac{d^2 W}{d\gamma_i d\gamma_k} = 0, \tag{12}$$

with:

$$D^2_{i,k}\Psi = \frac{\partial}{\partial\gamma}\left(\frac{\partial\Psi}{\partial\gamma}e_i\right)e_k + \frac{\partial}{\partial W}\left(\frac{\partial\Psi}{\partial\gamma}e_i\right)\frac{dW}{d\gamma_k} + \frac{\partial}{\partial W}\left(\frac{\partial\Psi}{\partial\gamma}e_k\right)\frac{dW}{d\gamma_i} + \frac{\partial}{\partial W}\left(\frac{\partial\Psi}{\partial W}\frac{dW}{d\gamma_i}\right)\frac{dW}{d\gamma_k}.$$

If we substitute the second-order derivatives of the state with respect to uncertain parameters $\frac{d^2W}{d\gamma_i d\gamma_k}$ in equation (11) from equation (12), we get:

$$\frac{d^2j}{d\gamma_i d\gamma_k} = D^2_{i,k}J - \frac{\partial J}{\partial W}\left(\frac{\partial\Psi}{\partial W}\right)^{-1} D^2_{i,k}\Psi \tag{13}$$

$$= D^2_{i,k}J - \Pi^\top D^2_{i,k}\Psi, \tag{14}$$

where Π is the solution of the adjoint system (9). This approach was firstly proposed by [SHE 96] and is usually known as *Tangent on Tangent* (ToT) approach or *Forward-on-Forward* ([GHA 07]), since we apply two direct differentiations to the functional.

0.3.3. *Algorithm*

The algorithm to compute the first and second derivatives is finally summarized in the next table :

1. Solve for the adjoint variables Π in: $\left(\frac{\partial\Psi}{\partial W}\right)^\top \Pi = \left(\frac{\partial J}{\partial W}\right)^\top$
2. Compute the gradient of j: $\left(\frac{dj}{d\gamma}\right)^\top = \left(\frac{\partial J}{\partial\gamma}\right)^\top - \left(\frac{\partial\Psi}{\partial\gamma}\right)^\top \Pi$
3. For $i \in 1 \ldots n$: Solve for the flow sensitivities θ_i in: $\left(\frac{\partial\Psi}{\partial W}\right)\theta_i = -\left(\frac{\partial\Psi}{\partial\gamma_i}\right)$
4. For each $i \in 1 \ldots n$ and $k \in 1 \ldots i$, compute: $\frac{d^2j}{d\gamma_i d\gamma_k} = D^2_{i,k}J - \Pi^\top (D^2_{i,k}\Psi)$

0.3.4. *Use of automatic differentiation*

In order to obtain the terms that appear in the algorithm above and containing the first- and second-order derivatives, we need a differentiated version of the original CFD code and this differentiation, if performed "by hand" is tedious and error-prone.

Then, we prefer to compute them using the automatic differentiation (AD) software Tapenade [HAS 04], developed by Tropics Project-Team at INRIA. This software is used to generate automatically a source code that computes the derivatives of an original FORTRAN code.

Consider a program that computes an output vector $v \in \mathrm{R}^m$ from an input vector $u \in \mathrm{R}^n$ as a function $v = \phi(u)$. The derivative of the function is provided by the Jacobian matrix $\frac{\partial \phi}{\partial u}$. The program is a sequence of elementary instructions that can be identified with a composition of elementary functions. The AD tool simply applies the chain rule to differentiate these elementary functions to obtain the desired Jacobian matrix. However, we are usually not interested by the knowledge of the full Jacobian matrix. Then, Tapenade has two differentiation modes, that allow to compute the product of the Jacobian matrix by a given vector. We can perform this matrix-by-vector product in a twofold manner: by right (*tangent mode*) or by left (*reverse mode*):

– the tangent mode allows to compute, from an arbitrary direction $\dot{u} \in \mathrm{R}^n$, the derivative in the direction $\dot{u}$:

$$u, \dot{u} \mapsto \left.\frac{\partial \phi}{\partial u}\right|_u \dot{u}$$

– the reverse mode allows to compute, from an arbitrary direction $\bar{\phi} \in \mathrm{R}^m$, the following product:

$$u, \bar{\phi} \mapsto \left(\left.\frac{\partial \phi}{\partial u}\right|_u\right)^\top \bar{\phi}$$

These two modes can be employed to easily compute the terms that are required for derivatives estimation.

Consider that the functional $j = J(\gamma, W)$ is computed in a FORTRAN subroutine `func`, whose input variables are `gamma` and `w` and output variable is `j`:

$$\underset{\downarrow}{\texttt{func}}(\underset{J}{\texttt{j}}, \texttt{gamma}, \texttt{w}).$$

If we perform a reverse mode differentiation with respect to the input variables `gamma` and `w`, we obtain a new subroutine:

$$\texttt{func\_b}(\underset{J}{\texttt{j}}, \texttt{jb}, \texttt{gamma}, \texttt{gammab}, \texttt{w}, \texttt{wb}),$$

where `jb` a new input variable and `gammab` and `wb` are new output variables defined as:

$$\bar{\gamma} = \left(\frac{\partial J}{\partial \gamma}\right)^{\top} \bar{J} \qquad \bar{W} = \left(\frac{\partial J}{\partial W}\right)^{\top} \bar{J}. \tag{15}$$

If we evaluate the above subroutine with the input $\bar{J} = 1$, the quantities in (15) are the first term in the right hand side (r.h.s.) of (10) and the r.h.s. of the adjoint equation (9).

Now consider that the subroutine that computes the state residuals $\Psi(\gamma, W)$ is `state`, whose input variables are `gamma` and `w` and output variable is `psi`:

$$\underset{\downarrow\,\Psi}{\texttt{state}}(\underset{\downarrow}{\texttt{psi}}, \texttt{gamma}, \texttt{w}).$$

If we perform a tangent mode differentiation, we can easily compute the product of the derivatives of the state residuals with respect to state variables $\frac{\partial \Psi}{\partial W}$ with a given vector, or the derivatives of the state residuals with respect to the uncertain parameters $\frac{\partial \Psi}{\partial \gamma_i}$, required to compute the flow sensitivities in (8). For the first quantity we need to differentiate with respect to the input variable `w`, namely:

$$\underset{\downarrow\,\Psi \quad \downarrow\,\dot{\Psi}}{\texttt{state\_dw\_d}}(\underset{\downarrow}{\texttt{psi}}, \underset{\downarrow}{\texttt{psid}}, \texttt{gamma}, \texttt{w}, \texttt{wd}),$$

where the new output variable `psid` contains the directional derivative $\dot{\Psi} = \frac{\partial \Psi}{\partial W}\dot{W}$. Similarly, for $\frac{\partial \Psi}{\partial \gamma_i}$ we need to differentiate with respect to the input variable `gamma`:

$$\underset{\downarrow\,\Psi \quad \downarrow\,\dot{\Psi}}{\texttt{state\_dgamma\_d}}(\underset{\downarrow}{\texttt{psi}}, \underset{\downarrow}{\texttt{psid}}, \texttt{gamma}, \texttt{gammad}, \texttt{w}),$$

and now the new output variable `psid` is $\dot{\Psi} = \frac{\partial \Psi}{\partial \gamma}\dot{\gamma}$. Thus, to compute $\frac{\partial \Psi}{\partial \gamma_i}$ is sufficient to set $\dot{\gamma} = e_i$, where e_i is the i-th vector of the canonical basis.

If we perform a reverse mode differentiation with respect to the input variables, we obtain the following new subroutine:

$$\underset{\downarrow\,\Psi \qquad\qquad\qquad\qquad\qquad\qquad \downarrow\,\bar{\gamma} \qquad \downarrow\,\bar{W}}{\texttt{state\_b}}(\underset{\downarrow}{\texttt{psi}}, \underset{\downarrow}{\texttt{psib}}, \texttt{gamma}, \texttt{gammab}, \texttt{w}, \texttt{wb}),$$

where `gammab` and `wb` are new output variables and `psib` a new input variable. A call to this subroutine allows to compute the matrix-by-vector products:

$$\bar{\gamma} = \left(\frac{\partial \Psi}{\partial \gamma}\right)^\top \bar{\Psi} \quad \bar{W} = \left(\frac{\partial \Psi}{\partial W}\right)^\top \bar{\Psi}. \tag{16}$$

A similar subroutine (without the $\bar{\gamma}$ term) is used to solve the adjoint equation (9). Indeed, using an iterative matrix-free linear solver (e.g. GMRES) in which the matrix-by-vector products are obtained by the differentiated routine (i.e. the $\bar{W}$ term in (16)), we can easily solve the linear system (9) without to store the Jacobian $\left(\frac{\partial \Psi}{\partial W}\right)^\top$.

To compute the terms required to estimate the second-order derivatives (14), we need to perform two successive tangent-mode differentiations. For example, considering the tangent-mode differentiation of the subroutine `func` with respect to `gamma` and `w` we obtain:

$$\underset{\substack{\downarrow \\ J}}{\texttt{func\_d}}\underset{\substack{\downarrow \\ \dot{J}}}{(\texttt{j}}, \underset{\substack{\downarrow \\ \dot{\texttt{j}}}}{\texttt{jd}}, \underset{\downarrow}{\overset{\gamma}{\texttt{gamma}}}, \underset{\downarrow}{\overset{\dot{\gamma}}{\texttt{gammad}}}, \underset{\downarrow}{\overset{W}{\texttt{w}}}, \underset{\downarrow}{\overset{\dot{W}}{\texttt{wd}}}),$$

where `gammad` and `wd` are new input variables and `jd` a new output variable. These input variables are provided by the user, whereas the output variable is the directional derivative:

$$\dot{J} = \frac{\partial J}{\partial \gamma}\dot{\gamma} + \frac{\partial J}{\partial W}\dot{W}.$$

Then, the differentiation of the output variable `jd` in the subroutine `func_d` with respect to input variables `gamma` and `w` gives us:

$$\underset{\substack{\downarrow \\ J}}{\texttt{func\_dd}}\underset{\substack{\downarrow \\ \dot{J}}}{(\texttt{j}}, \underset{\substack{\downarrow \\ \dot{\texttt{j}}}}{\texttt{jd}}, \underset{\substack{\downarrow \\ \ddot{\texttt{j}}}}{\texttt{jdd}}, \underset{\downarrow}{\overset{\gamma}{\texttt{gamma}}}, \underset{\downarrow}{\overset{\dot{\gamma}_0}{\texttt{gammad0}}}, \underset{\downarrow}{\overset{\dot{\gamma}}{\texttt{gammad}}}, \underset{\downarrow}{\overset{W}{\texttt{w}}}, \underset{\downarrow}{\overset{\dot{W}_0}{\texttt{wd0}}}, \underset{\downarrow}{\overset{\dot{W}}{\texttt{wd}}}),$$

where `jdd` is the new output variable, that represents:

$$\begin{aligned} \ddot{J} = &\frac{\partial}{\partial \gamma}\left(\frac{\partial J}{\partial \gamma}\dot{\gamma}\right)\dot{\gamma}_0 + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial \gamma}\dot{\gamma}\right)\dot{W}_0 \\ &+ \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial \gamma}\dot{\gamma}_0\right)\dot{W} + \frac{\partial}{\partial W}\left(\frac{\partial J}{\partial W}\dot{W}\right)\dot{W}_0. \end{aligned} \tag{17}$$

If one calls this subroutine with the following input parameters:

$$\dot{\gamma}_0 = e_k \quad \dot{\gamma} = e_i \quad \dot{W}_0 = \frac{dW}{d\gamma_k} \quad \dot{W} = \frac{dW}{d\gamma_i}, \tag{18}$$

one obtains as output variable $\dot{\dot{J}} = D^2_{i,k}J$. Then this routine is used to compute the second order derivatives according to (14). Note that the flow sensitivities $\dot{W} = \frac{dW}{d\gamma}$ should be computed and stored before the evaluation of $D^2_{i,k}J$, according to the algorithm presented previously. The term $D^2_{i,k}\Psi$ can be computed in the same way, by applying a tangent differentiation mode to the subroutine `state_d`. As conclusion, AD can be used in an efficient way to compute the first and second derivatives of the functional, according to the algorithm presented in a previous section.

0.4. Metamodel-based Monte-Carlo method

0.4.1. *Presentation*

An alternative and straightforward approach to estimate statistical quantities for the performance is to employ a classical Monte-Carlo method. The Monte-Carlo methods have already been presented and discussed in a previous chapter, then we only remind here the main features. It consists in generating a sample of the uncertain parameters $(\gamma^i)_{i=1,\dots,N_{MC}}$ and then estimate the expectation and the variance of the performance by using an unbiased predictor, such as:

$$\hat{\mu}_j = \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} j(\gamma^i) \tag{19}$$

and

$$\hat{\sigma}_j = \frac{1}{N_{MC}-1} \sum_{i=1}^{N_{MC}} (j(\gamma^i) - \hat{\mu}_j)^2. \tag{20}$$

However, it is well known that the accuracy of such an estimation depends critically on the sample size N_{MC}. As consequence, a large number of simulations should be performed for a single estimation of the performance statistics. For typical engineering applications, a sample size of some hundreds is required for an accurate estimation of the expectation, whereas a sample size of several thousands is required for a satisfactory estimation of the variance. Therefore, it is usually not possible to employ such a strategy by using simulation codes directly to compute the system performance.

Nevertheless, Monte-Carlo estimation can be used on the basis of an approximated model, that replaces the expensive PDEs solving procedure. Metamodels can be employed for this purpose. Metamodels are constructed according to available data that are stored in a database. It consists in using these data (performance already computed for some parameters values) to predict the performance for new parameters values. This database can be generated separately or compiled during an optimization procedure for instance. Metamodels mostly used for data fitting are:

– polynomial fitting (least-squares approximation) ;
– artificial neural networks (multi-layer perceptrons) [RIE 94];
– radial basis functions [POW 01] ;
– Kriging methods (Gaussian process models) [SAC 89].

The last three options are well suited to highly non-linear behaviors, such as those encountered in mechanics. Since these metamodels have already been discussed in a previous chapter, we will not describe them and we will focus on their application for uncertainty quantification.

0.5. Application to aerodynamics

0.5.1. *A subsonic flow example*

0.5.1.1. *Testcase description*

The testcase considered here corresponds to the flow around the wing of a business aircraft (courtesy of Piaggio Aero Ind.), for a subsonic regime. The flow analysis is performed by resolving the three-dimensional compressible Euler equations using a finite-volume CFD code developed at INRIA[DER 92].

The nominal operating conditions are defined by the free-stream Mach number $M_\infty = 0.65$ and the incidence $\alpha = 2^\circ$. The wing section is supposed to correspond to the NACA 0012 airfoil. The wing shape and the mesh in the symmetry plane are depicted in figure (1). The mesh employed counts 31124 nodes. We suppose that the free-stream Mach number and the angle of attack are subject to random fluctuations. For the sake of simplicity, we assume that their PDFs are Gaussian and uncorrelated. They are characterized by :

	Mach	Incidence (deg.)
Mean	0.65	2
Standard deviation	0.01666	0.03333

We aim at using the methods presented above to estimate the statistics on the drag coefficient, that is considered as the performance of the system.

0.5.1.2. *Reference results*

We compute first some reference results, obtained by performing 21×21 CFD analyses on a regular grid, as seen in figure (2). Reference statistical values are computed by constructing a fine metamodel based on these 21×21 points and performing a Monte-Carlo analysis. The following reference values for the mean and variance of the drag coefficient are obtained:

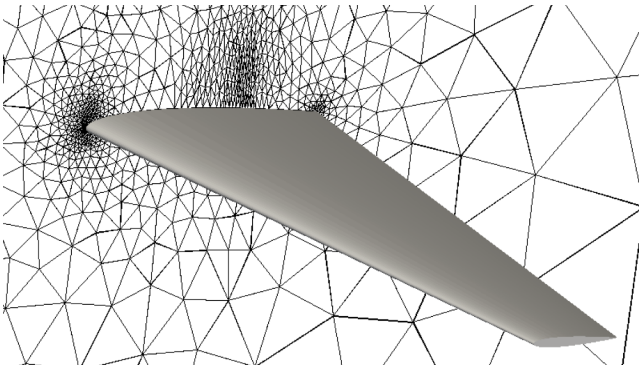

Figure 1. *Wing shape and mesh in the symmetry plane.*

Reference expectation	$\mu_j = 6.857\,10^{-3}$
Reference variance	$\sigma_j^2 = 1.553\,10^{-7}$

We have verified that these values do not depend on the choice of the metamodel (either RBFs or Kriging), and are not modified if extra points are added or if larger samples are used for the Monte-Carlo simulation.

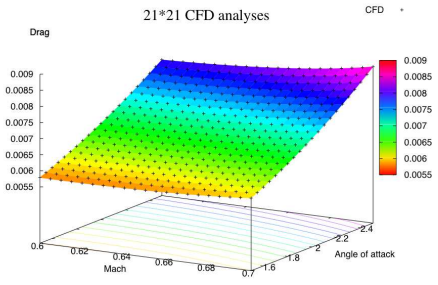

Figure 2. *Drag for* 21×21 *CFD analyses.*

0.5.1.3. *Results with meta-models (regular grid)*

We construct RBF and kriging meta-models for different database sizes that correspond to regular grids 3×3, 4×4 and 5×5. Then, Monte-Carlo simulations based on these meta-models are performed. The results in terms of drag expectation, variance and errors with respect to reference results are summarized in the following

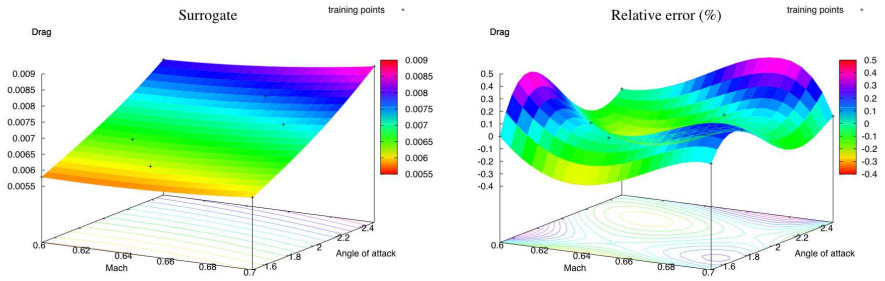

Figure 3. *Drag value and error in % using a RBF meta-model with 8 points (LHS sampling).*

tables:

Metamodel	Points	Expectation	Expectation error
RBF	4	$7.09309\,10^{-3}$	$2.35579\,10^{-4}$
RBF	9	$6.85813\,10^{-3}$	$6.21914\,10^{-7}$
RBF	25	$6.85756\,10^{-3}$	$5.03367\,10^{-8}$
KRG	4	$7.09309\,10^{-3}$	$2.35579\,10^{-4}$
KRG	9	$6.85932\,10^{-3}$	$1.80872\,10^{-6}$
KRG	25	$6.85756\,10^{-3}$	$5.44201\,10^{-8}$

Metamodel	Points	Variance	Variance error
RBF	4	$1.83171\,10^{-7}$	$2.78367\,10^{-8}$
RBF	9	$1.57570\,10^{-7}$	$2.23580\,10^{-9}$
RBF	25	$1.56382\,10^{-7}$	$1.04780\,10^{-9}$
KRG	4	$8.08668\,10^{-10}$	$1.54525\,10^{-7}$
KRG	9	$1.77749\,10^{-7}$	$2.24148\,10^{-8}$
KRG	25	$1.56307\,10^{-7}$	$9.72773\,10^{-10}$

As can be seen, results converge quickly as the size of the database increases.

0.5.1.4. *Results with meta-models (LHS)*

Then we construct RBF and kriging meta-models for databases generated by latin hypercube sampling. The databases also include the corners of the variation domain. Results are slighty less accurate for the mean estimate, but not for the variance estimate. Anyway, the difference is not significative for practical applications. Figure (3)

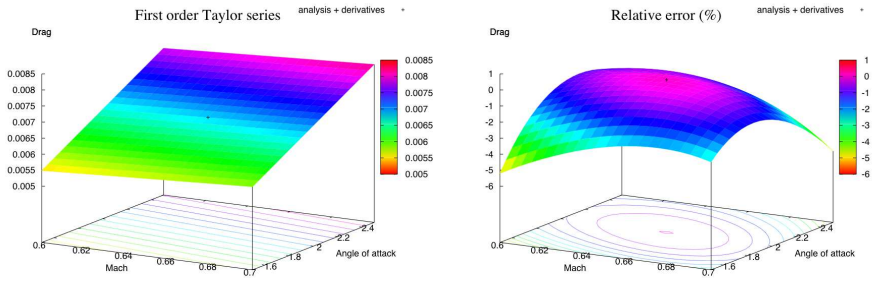

Figure 4. *Drag value and error in % using AD (first-order).*

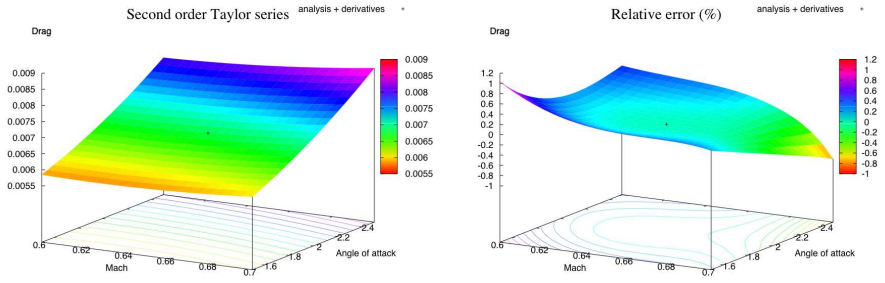

Figure 5. *Drag value and error in % using AD (second-order).*

shows the drag evolution with respect to the uncertain parameters obtained for a RBF meta-model with 8 points, as well as the error computed at 21×21 points. As seen, the error on the drag is less than 0.5%.

Metamodel	Points	Expectation	Expectation error
RBF	8	$6.85507\,10^{-3}$	$2.43491\,10^{-6}$
RBF	23	$6.85762\,10^{-3}$	$1.05410\,10^{-7}$
KRG	8	$6.85257\,10^{-3}$	$4.94454\,10^{-6}$
KRG	23	$6.85762\,10^{-3}$	$1.14150\,10^{-7}$

Metamodel	Points	Variance	Variance error
RBF	8	$1.56176\,10^{-7}$	$8.41631\,10^{-10}$
RBF	23	$1.56595\,10^{-7}$	$1.26098\,10^{-9}$
KRG	8	$1.55076\,10^{-7}$	$2.57555\,10^{-10}$
KRG	23	$1.56575\,10^{-7}$	$1.24084\,10^{-9}$

0.5.1.5. *Results with AD*

AD is then used to estimate drag statistics. Contrary to the previous case, the flow is only computed at the mean values of the uncertain parameters, as well as the derivatives. The statistics obtained using first- and second-order Taylor series are given in the next tables:

	Expectation	Expectation error
First-order	$6.83049\,10^{-3}$	$2.70266\,10^{-5}$
Second-order	$6.86056\,10^{-3}$	$3.04649\,10^{-6}$

	Variance	Variance error
First-order	$1.54665\,10^{-7}$	$6.69276\,10^{-10}$
Second-order	$1.55758\,10^{-7}$	$4.23060\,10^{-10}$

The accuracy of the results is similar to the one obtained with metamodels with 8 or 9 points. Figures (4) and (5) show the drag evolution with respect to the uncertain parameters obtained using AD, as well as the error computed at 21×21 points. One can observe that the error is larger at the corners than using meta-models. However, this is not critical for statistics estimation, since the PDFs of uncertain parameters become smaller and smaller as one moves away from nominal operational conditions.

0.5.1.6. *Comparison of computational performance*

Since the two proposed methods are essentially different, it is interesting to compare also their computational performance, in terms of CPU time and memory requirements. The following table details the memory used by the AD-based approach:

	Memory in Mb
Flow solver	130
First derivatives	250
Second derivatives	120
GMRES linear solver	250
Preconditionners	340
Total	1090

As seen, the computation of the flow solution with the first and second derivatives requires about 10 times more memory than the flow solution alone. However, this result can be improved using dynamic memory allocation or advanced programming tricks.

The computational time required is given in the next table:

	CPU time in second
Flow solver	403
Gradient	255
Hessian	278
Total	936

These results are obtained with an Intel Xeon 2.66 GHz. The AD-based approach is particularly efficient in this case, since the CPU time only increases about twice to obtain the gradient and Hessian required to compute the statistics.

Concerning the method based on meta-models, the costs are mainly related to the construction of the database. If it is built sequentially, the memory required is the same as the one used by the flow solver alone, whereas the CPU time increases linearly. If it is built using parallel computing, with a number of processors equal to the database size, the CPU time remains more or less similar to a single flow solver run. For instance, we obtain for a database with 8 points:

	CPU time in second
Sequential database	3250
Parallel database	440

In conclusion, the method based on meta-models is more expensive in terms of CPU time, except if one has the capability to build it using parallel computing.

0.5.2. *A transonic flow example*

0.5.2.1. *Testcase description*

We consider now a similar testcase, in transonic regime, with a wing shape that has been optimized for these particular flow conditions. More precisely, the free-stream Mach number is now set to $M_\infty = 0.83$ and the incidence $\alpha = 2°$. The wing planform remains unchanged, but the wing section has been optimized to minimize the drag under a constant lift constraint, for these flow conditions. In this case, only the free-stream Mach number is supposed to be subject to uncertainty (with Gaussian PDF), with:

	Mach	Incidence (deg.)
Mean	0.83	2
Standard deviation	0.01666	0

The evolution of the drag coefficient with respect to the Mach number is depicted in figure (6). As can be seen, it is more complex than in the previous exercise. The drag tends to be constant until $M_\infty = 0.83$ (optimization point) and then increases quickly. This behaviour corresponds to the generation of a strong shock wave, as soon as the Mach number is higher than that used during optimization.

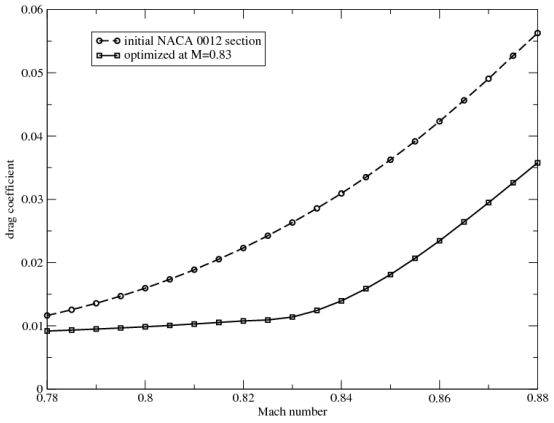

Figure 6. *Drag variation for fluctuating Mach number: initial design and optimum design at nominal Mach number (0.83).*

As previously, reference results are first obtained using 21 CFD analyses. Then, we test the uncertainty quantification methods to estimate the expectation and the variance of the drag. Figure (7) shows the results obtained using a linear or quadratic approximation, in terms of PDF of the drag. Both results are of poor quality. This is not surprising, since the drag evolution is far from a linear or even quadratic function. The linear model has obviously a poor accuracy and the resulting PDF is Gaussian. This is far from the reference result, for which the PDF has a more complex shape and is characterized by a peak at low drag values. The quadratic model is closer to the CFD calculations for high Mach numbers. Then, the tail of the PDF is quite well reproduced. However, the peak description is not satisfactory.

Then, Radial Basis Functions (RBF) are employed using five and seven training points (see figures (8) and (9)). As can be observed, better results are obtained. The metamodel built using five points exhibits a good agreement with reference results, except for the peak of the PDF. This is due to the discrepancy that can be observed between the RBFs fitting and the CFD results at Mach number 0.82. To accurately

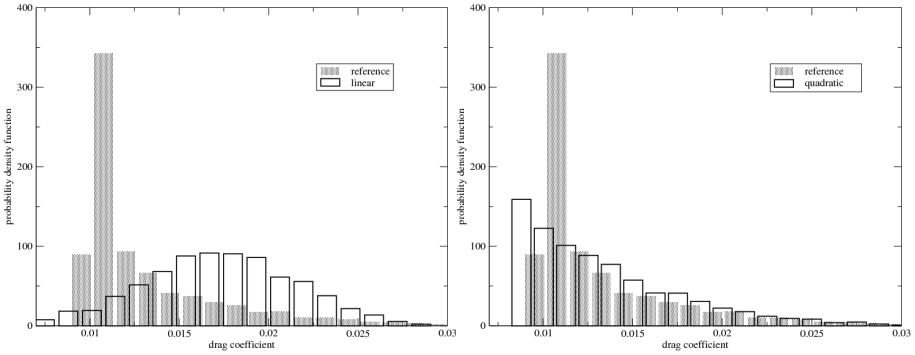

Figure 7. *Probability density function of the drag coefficient: reference result compared to linear and quadratic approximation.*

represent the curvature in this region, the database must be enlarged. Using seven training points, this defect is corrected and a satisfactory PDF prediction is obtained. Finally, the expectation and variance values corresponding to these different tests are provided in table (1).

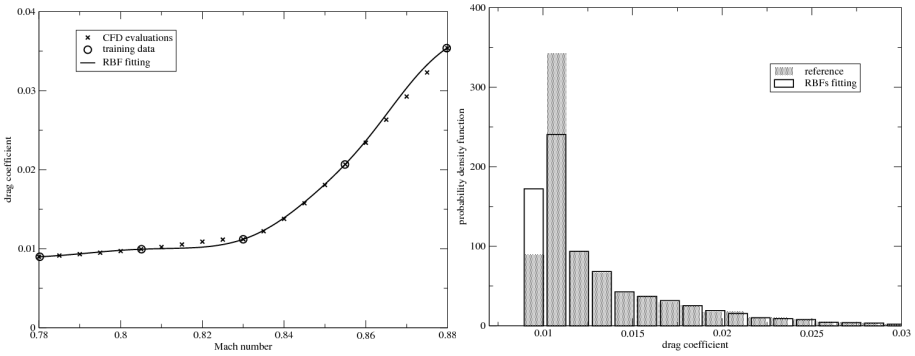

Figure 8. *RBFs using five training points : drag coefficient evolution and probability density function.*

This second exercise shows that the Taylor series approximation, linear or quadratic, can provide poor results when the performance evolution is characterized by high-order variations. This is especially the case when a system is optimized for some particular operating conditions. Metamodels can provide results of better accuracy, but for an increased computational cost.

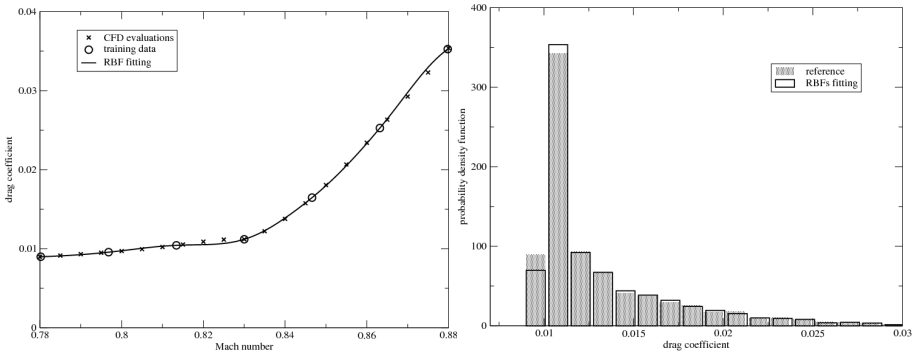

Figure 9. *RBFs using seven training points : drag coefficient evolution and probability density function.*

Case	Mean	Variance
Reference	0.013154	1.5658E-05
Linear	0.017229	1.7953E-05
Quadratic	0.013262	1.9482E-05
RBFs (5 pts)	0.013029	1.7229E-05
RBFs (7 pts)	0.013068	1.5899E-05

Table 1. *Statistics for the drag obtained with the different methods.*

0.6. Conclusion

In this chapter, we have presented two practical approaches that can be used to quantify uncertainty for systems governed by partial differential equations, in an industrial context. These two approaches rely on the construction of a simplified model to explore the vicinity of nominal operational conditions. The first one is based on a Taylor series expansion from the nominal point and requires the computation of derivatives. The second one uses metamodeling techniques for which the evaluation of the performance at several points is mandatory.

Both methods suffer from severe limitations at the present time. The method based on Taylor series is limited by the domain of validity of the Taylor expansion. Moreover, the computation of derivatives can be cumbersome for complex codes, even if AD softwares are useful. The source code should be available, this is usually not the case for commercial CFD or CSM codes. Finally, another difficulty arises from the fact that some parts of programs can be non-differentiable (e.g. programs with MIN or MAX operators). Concerning the second method, the main limitation is related to the increase of the database points required to build accurate metamodels in spaces

of high dimensions. As consequence, the number of uncertain parameters that can be considered with this approach remains limited.

Polynomial Chaos (PC) methods[KNI 06] are a possible alternative and are subject to an intense research activity. They can be implemented in two ways: intrusive and non-intrusive. In the former case, new simulation codes have to be developed using the PC formalism, yielding a difficult task in an industrial framework. The latter case allows the use of existing codes as black-boxes, but this approach is far more expensive and becomes more or less similar to a direct integration approach using numerical quadratures.

Therefore, uncertainty estimation for industrial problems, including a large number of uncertain parameters, still remains an issue.

0.7. Bibliography

[BEY 07] BEYER H.-G.SENDHOFF B., Robust optimization – A comprehensive survey, *Comput. Methods Appl. Mech. Engrg.*, vol. 196, p. 3190-3218, 2007.

[DER 92] DERVIEUX A.DÉSIDÉRI J.-A., Compressible flow solvers using unstructured grids, INRIA Research Report 1732, June 1992.

[GHA 06] GHATE D.GILES M. B., *Inexpensive Monte Carlo uncertainty analysis*, p. 203–210, Recent Trends in Aerospace Design and Optimization, Tata McGraw-Hill, New Delhi, 2006.

[GHA 07] GHATE D.GILES M. B., Efficient Hessian Calculation Using Automatic Differentiation, *25th Applied Aerodynamics Conference, Miami (Florida)*, 2007-4059, AIAA, June 2007.

[HAS 04] HASCOËT L.PASCUAL V., TAPENADE 2.1 user's guide, 0300, INRIA, Sep 2004.

[KNI 06] KNIO O.MAITRE O. L., Uncertainty propagation in CFD using polynomial chaos decomposition, *Fluid Dynamics Research*, vol. 38, 9, p. 616–640, September 2006.

[MAR 07] MARTINELLI M., Sensitivity Evaluation in Aerodynamic Optimal Design, PhD thesis, Scuola Normale Superiore (Pisa) - Université de Nice-Sophia Antipolis, 2007.

[POW 01] POWELL M., Radial basis function methods for interpolation to functions of many variables, *Fifth hellenic-European conference on Computer Mathematics and its applications*, 2001.

[PUT 01] PUTKO M. M., NEWMAN P. A., TAYLOR III A. C.GREEN L. L., Approach for uncertainty propagation and robust design in CFD using sensitivity derivatives, 2528, AIAA, 2001.

[RIE 94] RIEDMILLER M., Advanced supervised learning in multi-layer perceptron - from backpropagation to adaptative learning algorithms, *Computer standards and interfaces*, , 5, 1994.

[SAC 89] SACKS J., WELCH W., MITCHELL T.WYNN H., Design and analysis of computer experiments, *Statistical Science*, vol. 4, 4, p. 409–435, 1989.

[SHE 96] SHERMAN L. L., TAYLOR III A. C., GREEN L. L.NEWMAN P. A., First and second-order aerodynamic sensitivity derivatives via automatic differentiation with incremental iterative methods, *Journal of Computational Physics*, vol. 129, p. 307-331, 1996.

[Tay 01] TAYLOR III A. C., GREEN L. L., NEWMAN P. A.PUTKO M. M., Some advanced concepts in discrete aerodynamic sensitivity analysis, 2529, AIAA, 2001.